dit
UPM
Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación
Departamento de Ingeniería de Sistemas Telemáticos

EXAMEN FINAL ORDINARIO DE SISTEMAS OPERATIVOS
Convocatoria de Febrero. 9–2–99
PRIMERA PARTE (SIN LIBROS)

Puntuación y longitud *máxima* —puede de hecho ser más corta— de la respuesta, en cada pregunta. **Una página es una cara**.
DURACIÓN: 1 hora

Pregunta 1

Explique brevemente qué es un multienvío atómico, diga por qué, para realizarlo, no basta confirmar la entrega a todos los destinatarios y esboce algún algoritmo correcto.

(*1/2 página; 1 punto*)

Aquél en que la entrega se hace a todos los destinatarios que no fallan o a ninguno. No basta confirmar la entrega porque el que envía se puede caer a medias. Puede solucionarse haciendo que todos los que reciban un multienvío lo reenvien a su vez a todos. Menos costoso es vigilar al que envía, reenviando si se ve que se ha caído, para lo cual es necesario guardarse una historia de mensajes, que debería irse limpiando cuando se sepa que han sido recibidos por todos.

Pregunta 2

Cite tres ejemplos de servicios distribuidos que utilicen la jerarquización para mejorar la escalabibilidad.

(*1/3 página; 1 punto*)

Son escalables el servicio de nombres DNS, repartiendo la autoridad de subdominios, el servicio de tiempos NTP, definiendo estratos y comunicando las máquinas con los estratos vecinos, y los servicios de ficheros distribuidos, como NFS, AFS, etc, al repartir directorios en diversas máquinas.

Pregunta 3

Explique qué es replicación optimista y cite algún sistema explicado en clase que la utilice.

(*1/3 página; 1 punto*)

Replicación optimista es aquella que prefiere dar servicio antes de garantizar la coherencia, prometiéndonos detectar las incoherencias en cuanto sea posible. CODA es un ejemplo de replicación optimista.

Pregunta 4

Explique brevemente qué es la compartición falsa en sistema de memoria compartida distribuida y diga cómo un sistema con coherencia débil puede librarse del problema de la compartición falsa, mientras que uno con coherencia secuencial no puede.

(*1/2 página; 1 punto*)

La compartición falsa tiene lugar en los sistemas de memoria compartida distribuida cuando dos procesos usan variables distintas en la misma página. Si ambos escriben mucho en sus respectivas variables, se produce trasiego excesivo. Esto no se puede evitar en sistemas de coherencia secuencial pura. Si se pueden anotar las variables como compartidas y distintas, el compilador puede asignarles páginas distintas, lo que normalmente no hacen los sistemas de coherencia secuencial puros. En sistemas con coherencia débil se puede permitir trabajar sobre copias distintas (modo de escritura compartida), salvaguardando las páginas originales y, en la operación de sincronización (por ejemplo, esperas en barrera), actualizar sólo las diferencias. Si al actualizar se detecta una intersección entre las variables modificadas, se produce un error.

Pregunta 5

En la técnica de memoria virtual basada en un paginador externo (final del capítulo 6 del libro), existe un servidor de páginas remoto —que realiza un trabajo similar al de un *disco de red*—, al que llegan peticiones de lectura o escritura de páginas desde el núcleo del sistema cliente. Y éste tiene convenientemente modificado su manejador de disco para redirigir las lecturas y escrituras de páginas que se le piden como resultado de fallos de páginas locales, no al controlador de un disco local, que es lo que se hace en sistemas centralizados, sino a ese servidor de páginas remoto, a través de la red.

Se pide explicar brevemente cómo el sistema anterior puede ver alterado mínimamente su software para implementar, sin ayuda de ningún hardware adicional, compartición controlada de ficheros —e incluso de memoria principal, pero de ésta no hablar— entre procesos que se encuentran ejecutando en nodos diferentes. Mencionar el mayor problema que podría producirse con la alteración propuesta.

(*1/4 página; 1 punto*)

Para compartir ficheros sin control no hace falta hacer nada nuevo, solo dejar que el servidor de páginas acepte peticiones simultáneas desde ma's de un proceso de la red, que estén en el mismo nodo o en nodos distintos, para (partes idénticas de) un mismo fichero. Ahora bien, el esquema de paginador externo permite un control muy flexible (a escala de página) de la compartición de ficheros. Es decir, que se pueden compartir unas páginas de un fichero para lectura, mientras que otras del mismo fichero se usan de forma exclusiva para escritura por alguno(s) de los procesos.

Y además, se puede hacer también que ese control de acceso sea dinámico, en el sentido de que, por ejemplo, si dos procesos comparten la misma página de un fichero, en un principio para lectura, cuando uno de ellos la escriba, el otro pueda ver los cambios producidos al mismo tiempo que el que escribe (como en la memoria compartida distribuida). Solo es necesario para ello que el que escribe lo haga en modo "write-through" hacia el servidor de páginas, y que éste se comporte como cliente de los demás núcleos informándoles del nuevo valor de la página o de que, simplemente, han de cambiar los bits de protección del marco correspondiente para impedir todo tipo de acceso hasta que se refresque.

Y es aquí, al comportarse el servidor como "cliente de sus propios clientes", donde, si no se toman las medidas oportunas, existe un grave riesgo de interbloqueo debido a circularidad de peticiones cliente-servidor.

dit
UPM
Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación
Departamento de Ingeniería de Sistemas Telemáticos

EXAMEN FINAL ORDINARIO DE SISTEMAS OPERATIVOS
Convocatoria de Febrero. 9–2–99
SEGUNDA PARTE (CON LIBROS)

Puntuación y longitud *máxima* —puede de hecho ser más corta— de la respuesta, en cada pregunta. **Una página es una cara**.
DURACIÓN: 1 hora y cuarto

Pregunta 1

El observador pasivo de una computación distribuida de dos procesos recibe notificaciones de sucesos con horas vectoriales:

(1,0) (3,2) (1,1) (2,0) (1,2) (4,4) (2,4) (2,3)

Se pide:

- Poner en secuencia, y en una línea, uno de los órdenes coherentes posibles de observación de esas horas vectoriales y, en la línea siguiente y alineados con las horas vectoriales, los valores del vector de contadores de sucesos según se van aceptando los sucesos (horas vectoriales) correspondientes por el observador pasivo.

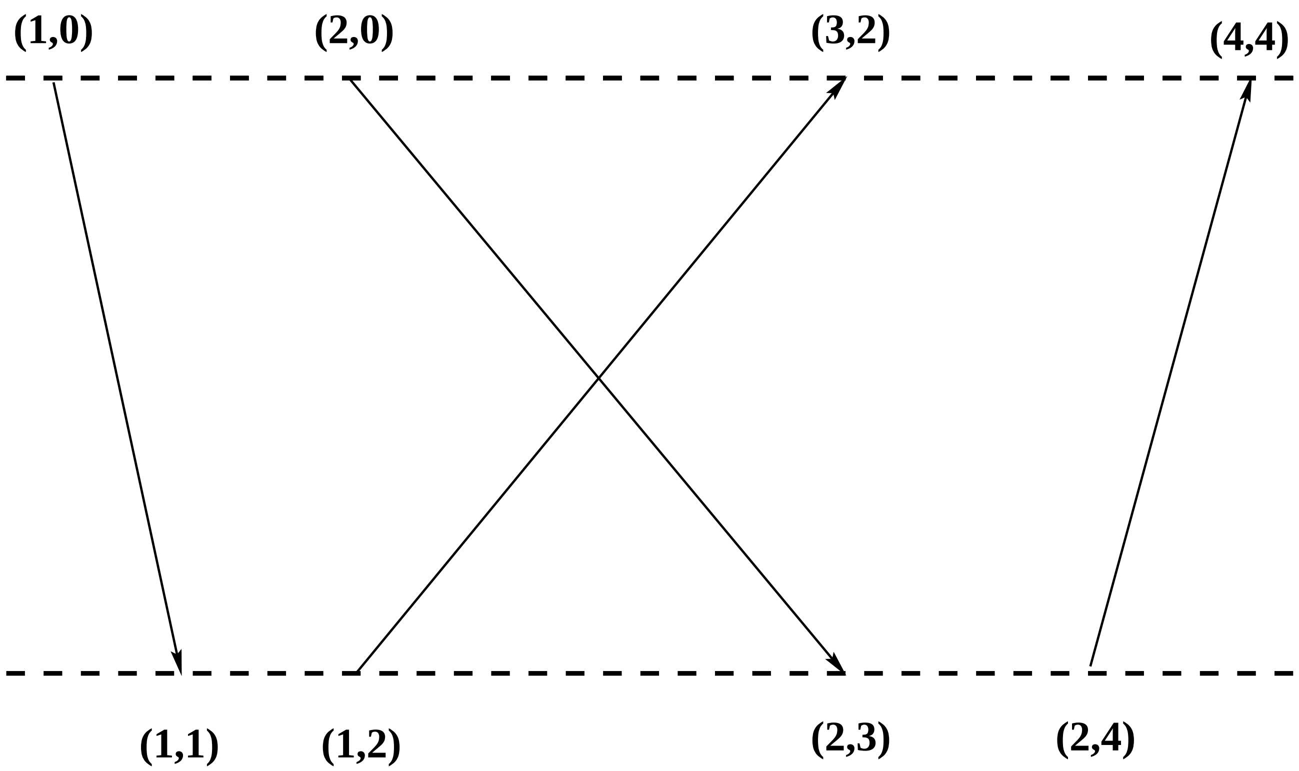
(*dos líneas; 1 punto*)

Hay muchas posibles, ésta es solo una de ellas:

(1,0)	*(1,1)*	*(1,2)*	*(2,0)*	*(3,2)*	*(2,3)*	*(2,4)*	*(4,4)*
[1,0]	*[1,1]*	*[1,2]*	*[2,2]*	*[3,2]*	*[3,3]*	*[3,4]*	*[4,4]*

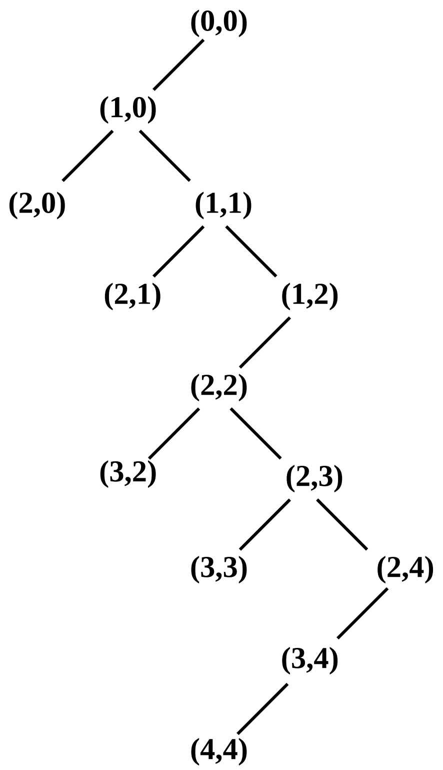
- A partir de la información anterior, construir ordenadamente el gráfico que especifica la computación distribuida en la manera usual, y poniendo cada una de las horas vectoriales al lado del suceso al que corresponde. (*Pista*: ir colocando los sucesos según el orden de aceptación por el observador, y descubrir cuándo en las recepciones de mensajes se presenta un "salto" en el contador correspondiente).

(*1/4 página; 1.5 puntos*)



- A partir del gráfico anterior, construir "a mano" (no hace falta utilizar las fórmulas, basta con observar las flechas que representan los mensajes) el retículo (*lattice*) de estados globales coherentes de la computación distribuida.

(*1/2 página; 1.5 puntos*)



Pregunta 2

En cierto sistema Linux, el administrador hace la siguiente operación de montaje de sistema de ficheros remoto NFS:

```
mount ftp.etsit.upm.es:/home/ftp/debian /mnt
```

y luego ejecuta el siguiente programa en C:

```
 1  #define TAMANO  100          /* Define el tamaño de las operaciones
 2                                  elementales de entrada/salida en octetos */
 3  #include <unistd.h>          /* Definiciones del sistema */
 4  #include <fcntl.h>
 5
 6  int main(void) {             /* Programa principal sin argumentos */
 7      int leidos, escritos;    /* Contadores enteros de octetos leídos y escritos */
 8      int origen, destino;     /* Descriptores de ficheros abiertos */
 9      char buf[TAMANO]         /* Zona de memoria de TAMANO octetos, para lecturas y escrituras */
10
11      /* Abre fichero en modo lectura */
12      origen = open("/mnt/hamm/Contents-i386.gz", O_RDONLY);
13
14      /* Abre fichero local para esctitura */
15      destino = open("/tmp/cosa", O_CREAT | O_WRONLY | O_TRUNC, 0644);
16
17      leidos = read(origen, buf, TAMANO);        /* Intenta leer hata TAMANO octetos de origen */
18      while (leidos >= 0) {                      /* Mientras lea más de 0 octetos */
19         escritos = write(destino, buf, leidos); /*   Escribe los octetos leidos en destino */
20         leidos = read(origen, buf, TAMANO);     /*   Intenta leer hata TAMANO octetos de origen */
21      }
22      close(origen);                             /* Cierra fichero origen */
23      close(destino);                            /* Cierra fichero destino */
24      exit(0);
25  }
```

Suponiendo que el montaje fuerce a que el intercambio de mensajes de NFS se haga en datagramas de 1100 caracteres de datos como máximo (sin contar cabeceras) y que el fichero que se abre en lectura existe y tiene 9000 caracteres. Diga:

- ¿Qué líneas del programa anterior generan tráfico y a qué operaciones de NFS corresponden? (ver figura 8.3 del libro)

  (*dos líneas; 0.5 puntos*)

  La línea 12 genera 2 operaciones de `lookup` *y las líneas 17 y 20 generan varias operaciones de lectura* `read`*.*

- ¿Cuántas operaciones de lectura de NFS tendrán lugar suponiendo que no haya procesos de carga anticipada (*bio-daemons*) y cúantas si los hay?

  (*tres líneas; 0.5 puntos*)

  No tiene que haber diferencias si el cliente hace peticiones de un bloque y luego va entregando los trozos necesarios. La única diferencia estará en los momentos en que se piden (antes en el caso de los bio-daemons). En ambos casos se empaquetarán las lecturas a 1100 caracteres, por lo que habrá solamente 9 operaciones de lectura remota. Por supuesto, suponemos que no hay bloques del fichero en la cache local.

  Si el cliente sin bio-daemons no empaquetara, cada lectura hará una petición, lo que daría lugar a 90 lecturas remotas. Sin embargo sería una realización muy pobre.